549576
P12

NASA
Technical
Paper
2957

November 1989

Fortran Program for X-Ray Photoelectron Spectroscopy Data Reformatting

Phillip B. Abel

NASA

[illegible]

NASA
Technical
Paper
2957

1989

Fortran Program for X-Ray Photoelectron Spectroscopy Data Reformatting

Phillip B. Abel
Lewis Research Center
Cleveland, Ohio

National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Division

Summary

A Fortran program has been written for use on an IBM PC/XT or AT or compatible microcomputer (personal computer, PC) that converts a column of ASCII-format numbers into a binary-format file suitable for interactive analysis on a Digital Equipment Corporation (DEC) computer running the VGS-5000 Enhanced Data Processing (EDP) software package. The incompatible floating-point number representations of the two computers were compared, and a subroutine was created to correctly store floating-point numbers on the IBM PC, which can be directly read by the DEC computer. Any file transfer protocol having provision for binary data can be used to transmit the resulting file from the PC to the DEC machine. The data file header required by the EDP programs for an x-ray photoelectron spectrum is also written to the file. The user is prompted for the relevant experimental parameters, which are then properly coded into the format used internally by all of the VGS-5000 series EDP packages.

Introduction

An increasing number of sophisticated surface analysis systems are today controlled by computers that provide the operator with not only an easier task in running the equipment but also, in some cases, with integral data analysis capabilities. The ability to perform the data analysis interactively with a color graphics display speeds the process considerably when compared with batch job processing. Interactive analysis also allows the researcher to watch for spurious data that might go undetected when some form of automatic spectrum processing is used.

The commercial data analysis package acquired by our research group was a VGS-5000 system running the Enhanced Data Processing (EDP) software. This software can perform peak synthesis, peak deconvolution, background subtraction, peak area measurement, data smoothing, differentiation and integration, spike removal, and satellite subtraction, as well as provide spectrum overlays, montage plots, and expanded viewing areas on the graphics terminals. The desirability of "importing" data from other similar experimental systems quickly became apparent. In particular, we had an older x-ray photoelectron spectroscopy (XPS) system that could digitally store data but had only limited data analysis software. Additionally, in order to test the performance of the data analysis routines, a means of creating controlled test data files was needed. This report details the methods used to create EDP-compatible files from externally generated data.

A recent publication (ref. 1) details the results of the Versailles Project on Advanced Materials and Standards (VAMAS) Surface Chemical Analysis group effort to produce the Standard Data Transfer Format. The need for a standard exchange format for surface analysis data is exemplified by the desirability of using a second computer to analyze data captured by a dissimilar computer. The VAMAS Standard Data Transfer Format uses only ASCII code characters (ref. 2) and provides for a wide range of surface analysis techniques. The type of information contained in the EDP file header closely parallels the VAMAS standards, although the information is stored in a more space-efficient binary form. Unfortunately, a program to convert ASCII format data to binary is not yet available for the EDP software. The program described herein is a limited start toward that goal for one analysis technique (XPS) and could be modified to parse for the VAMAS code words rather than requiring keyboard input of experimental parameter values.

Methods

Hardware and Communications

The computer that was purchased to run our data analysis software was a multiterminal, multiuser system with eight RS-232C serial ports. It had no programming language other than the assembly language native to the machine. In order to import data, any of the unused serial ports could have been connected to a modem or, as was done, connected directly to another computer via a null-modem serial cable. Direct connection avoids the potential for unauthorized computer access over the telephone lines, and in our case allowed the second computer to be used as another data analysis workstation because the short serial cable permitted a higher communication speed than is typically possible over telephone lines. Painting a screen full of graphics usually involves many more characters to be sent than does a simple display of text. The higher communication speed therefore made practical the use of graphics terminal emulation software on the second computer. A number of graphics terminal emulation programs exist for the IBM PC/XT or AT and compatible microcomputers (personal computers, PC), depending upon the graphics display hardware present. Whether or not they have graphics capability, most terminal emulation programs come with built-in data transfer protocols such as Kermit and XMODEM.

For file transfer, the set of public domain programs that implement the Kermit protocol has been chosen for use in our laboratory, in part because versions are available both for the Digital Equipment Corporation (DEC) PDP-11/73 running the Micro-RSX operating system as well as for the PC.[1] The protocol allows transmission of eight-bit binary (nontext) files, even over serial lines configured for seven-bit data. Although the Kermit program for the PC does not support emulation of DEC color graphics terminals and cannot be used with the EDP interactive data analysis routines, it does emulate the DEC VT102 terminal, which the Micro-RSX operating system recognizes. Additionally, all of the DEC color graphics terminal emulation programs known to the author do support file transfer via the Kermit protocol. If the PC were to be used as an interactive data analysis terminal, the Kermit protocol on the DEC machine would still be usable for file transfer. Other methods of binary file transfer, such as transfer over some form of local area network, can also be implemented, with potentially significant improvements in data transmission rate over that of a serial port. However, in most cases, the Kermit protocol should prove adequate.

Language

The Fortran language on the PC was chosen to reformat the external data for importation into the EDP software. Fortran remains a computer language familiar to most scientists and engineers and is available on microcomputers at many facilities. A Fortran compiler could be purchased for use on the DEC machine also. This would avoid the problem of incompatible floating-point number representations. In general, however, the prices of language compilers for small, single-user microcomputers are significantly less than those for multiuser computer systems, and a relatively simple, general-purpose subroutine solves the incompatible floating-point number problem.

Input Files

As written, the data-reformatting program uses an example data header file stored under the name TEMPLATE.XPS on the PC (see the appendix for an explanation of the data file header). An initial source for this file can be any EDP data file from an experiment similar to the type of data being imported. Downloading a binary data file from the DEC machine to the PC can serve also as a first check to determine whether the data transfer protocol is functioning properly. If for some reason it is not possible to download this file, another alternative (strongly discouraged by the author) is to use the information about the header in the appendix and the DEBUG.COM program on the PC to create the initial TEMPLATE.XPS data header file.

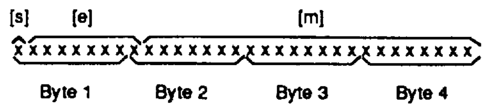
The input data file used by the program is no more than a text file in the form of a column of ASCII numbers, with each value followed by a carriage return. These can be the output of a data collection routine or can even be keyed in through the use of a program editor. By modifying the data-reading portion of this program, almost any input data format can be accommodated. For instance, the file from a spreadsheet "print to disk" operation could also be accepted. Once the PC has a value stored internally as a floating-point number, whether from text or even binary number input, the conversion from PC to DEC binary format is straightforward. As written, the program treats the first input file entry as the number of data values to follow in the file. The user is prompted for the rest of the spectrum parameters when the program is run.

Formats

Floating-Point Numbers

For creating files to be read unaltered by computers of different manufacturers with different operating systems and machine architectures, the first discussion should be of numeric representation formats. How do the binary digits, or bits, stored in a binary data file correspond to the values used by a program? A cursory discussion of the ANSI/IEEE Standard 754–1985 for 32-bit floating-point number representation (fig. 1) follows.

The most significant (left-most) bit gives the sign, with a 0 indicating a positive number and a 1 indicating a negative number. The next eight most significant bits are an exponential factor, or multiplier, for the fractional part of the number represented by the remaining 23 bits. In order to have both positive and negative exponents, the eight-bit exponent that is stored is offset by 127, which is approximately half of the maximum value an eight-bit number can have. The exponent



ANSI/IEEE: Value = $(-1)^{[s]} \cdot 2^{([e]-127)} \cdot [1.m]$

DEC: Value = $(-1)^{[s]} \cdot 2^{([e]-128)} \cdot [0.1m]$

Figure 1.—Thirty-two-bit floating-point number representations. Each "x" represents one binary digit, either a 0 or a 1. Numbers enclosed in square brackets are in binary form.

[1]For information about Kermit documentation, updates, lists of current available versions, and ordering information, write to Kermit Distribution, Columbia University Center for Computing Activities, 612 West 115th Street, New York, NY 10025 (USA).

factor can effectively range from 2^{-127} to 2^{128}. In order to use the exponent, the value 127 (the offset) must first be subtracted from the value actually stored. The mantissa, or fractional part of the floating-point number, is stored in a "hidden bit normalization" form. The mantissa is normalized by shifting the magnitude of the fractional part and then adjusting the value of the exponent until the most significant binary digit of the mantissa is a 1. This preserves maximum accuracy in representing the fractional part of the number by preventing leading zeros. Since the first bit of every fraction is a 1, there is no need to actually store that digit. This convention increases the accuracy by one bit. The leading bit is not assumed to be a 1 only when the exponent is 0. This corresponds to the smallest floating-point numbers that can be represented. There need not be concern about special cases for this discussion.

The differences between the IEEE and DEC PDP-11 floating-point representations are the offset used for the exponent and the assumed magnitude of the mantissa. DEC uses an exponent offset of 128 rather than 127, thereby giving the exponent a range from 2^{-128} to 2^{127}. This differs from the IEEE format by a factor of 2. The DEC format also assumes a mantissa of the form 0.1xxx..., where each "x" represents one of the 23 least significant binary digits of the 32-bit floating-point number (each "x" is either a 1 or a 0). The IEEE assumed representation is of the form 1.xxx.... The DEC format differs from the IEEE again by a factor of 2. In both representations, the leading 1 is assumed and need not be stored, leaving 23 bits to represent the rest of the mantissa, denoted here by x. In order to convert a number from the IEEE representation to the DEC form, both a shift in the implied mantissa value and an increase of the exponent must occur. Both changes can be accomplished by multiplying the IEEE representation by a factor of 4 to obtain the DEC representation.

The final floating-point number issue that must be addressed is the manner in which the numbers are stored in a file. If the 32 bits of a floating-point number are divided into four groups of eight bits each, or bytes, the most significant (leftmost) eight bits can be arbitrarily called "byte 1"; the next most significant eight bits (bits 23 through 16), "byte 2"; and so on (fig. 1). Apparently because of the internal architectures of the two types of machine, the most efficient method of storage for each machine involves a different byte ordering.

For the Intel 80x86 processor-chip based machines (8086, 8088, 80286, or 80386, i.e., IBM PC/XT or AT and compatibles), the first byte sequentially stored in a file is byte 4, followed by byte 3, byte 2, and finally, byte 1. This byte ordering is used in each of the languages checked by the author.[2]

The DEC machine, originally based on a 16-bit bus architecture, reverses every pair of bytes, storing them sequentially in the order: byte 2, byte 1, byte 4, and finally, byte 3. The difference between the two types of machine requires only that the pair of bytes 2 and 1 be shifted in front of the pair of bytes 4 and 3 for the change from 80x86 to PDP-11 byte ordering. In Fortran, an easy method of accomplishing the byte swap uses an EQUIVALENCE operation between a REAL*4 variable and an INTEGER*2 variable array. The two INTEGER*2 values are exchanged and the resulting modified REAL*4 variable can then be stored on disk. Because both machines reverse the byte order for two-byte numbers, standard 16-bit integers can be passed unaltered between the two types of computer.[3]

Output Files

In order to be used with the EDP software, an external data file must not only present readable numbers, but must also incorporate any file headers or other information expected by the analyzing routines. The following discussion is a short overview of the data file structure written and read by the EDP software on the VGS-5000 series of systems.

Files on the DEC PDP-11/73 computer are usually stored in units of "blocks," each 512 bytes in size. Each EDP data file has at least a four-block information section ahead of the actual data. The header information is needed because data files from a number of different experimental techniques, as well as multiregion and depth profile data, can be analyzed with the EDP software. The files produced by the routine reported here are in the form of a single-spectral-region, binding-energy-scan XPS spectrum. Comments are included in the source code, which should allow easy expansion of the program to certain other types of data file. Each block of the file header is detailed byte by byte in the appendix.

The first block of the file header provides general information about the data file, such as how many spectral regions are present and how large they are. The second block provides space for region names up to the maximum number of regions allowed, although only single-region files are created by this program. Up to three lines of descriptive comment can be stored in the third file header block. The fourth header block, describing the data immediately following it, contains information about the experimental technique and

[2]Binary floating-point number files from the following languages were available to the author: Lahey Fortran, Microsoft Fortran, Microsoft QuickBASIC 4.0, and Borland Turbo Pascal (on a PC with mathematic coprocessor chip installed). Turbo Pascal, in particular, may use its own 48-bit floating-point number representation on machines not equipped with the coprocessor chip. Therefore, care should be exercised if the algorithm presented herein is rewritten in Pascal. Fortran compilers not listed here presumably use the machine-efficient byte ordering called for by either an 8087 or an 80287 coprocessor, whether a coprocessor chip is present or not. However, the byte order of a test file should be checked before trying to use this program unmodified with other Fortran compilers on a PC.

[3]The program ASCITOVG is available from COSMIC, University of Georgia, Athens, GA.

conditions such as the type of scan, the range of the scan, the excitation source, and the analyzer mode. This header block would recur once for each spectral region in a multiregion file. It is located directly ahead of the data blocks that it describes and separates them from the data of preceding regions. The fifth and all subsequent blocks in a single-region file contain the four-byte data values, stored 128 to the block.

The Fortran program itself can be described in terms of a few basic functions. After checking for the header file TEMPLATE.XPS in the PC default directory, the program prompts the user for input and output file names to use (with or without path names) and opens the files. The number of input points is read, and the minimum and maximum input values in the file are found. The user is then prompted for information such as the starting and ending scan energies, the electron analyzer pass energy, the excitation source, the dwell time, and the number of combined (integrated) scans making up the data. The user is also asked for the time and date that the data were collected and is allowed to enter descriptive comments that will appear in plots of the spectrum. The actual reformatting of the data is then performed to finish the process.

Summary of Results

The incompatible floating-point number representations of a personal computer (PC) and a Digital Equipment Corporation (DEC) computer were compared, and a Fortran subroutine was created to correctly store single-precision, floating-point numbers on the PC that can be directly read by DEC computers. A Fortran program using this subroutine was written on the PC to convert a column of ASCII-format numbers into a binary-format file suitable for interactive analysis with the VGS-5000 Enhanced Data Processing (EDP) software package. More difficult than the reformatting of floating-point numbers was the creation of the exact data file header required by the EDP programs for an x-ray photoelectron spectrum. Experiment parameters, entered by the program user, are coded into the header format used internally by all of the VGS-5000 series EDP packages. Whether for externally captured data or for user-generated test data, the files created by the program described here should be useful on any of the VGS-5000 series interactive analysis workstations.

Interested VGS-5000 users may send the author a blank, formatted, 5.25-in. IBM PC/XT or AT compatible floppy diskette for a copy of the program source code, executable program file, and example header file. The program should also soon become available through the COSMIC software service operated for NASA by the University of Georgia (call (404) 542-3265 for further information).

Acknowledgments

The author would like to thank Daniel L. Whipple of NASA Lewis for providing a set of data-handling subroutines useful in transferring data between any of the many machines at NASA Lewis, including Cray, IBM, VAX, Alliant, Sun, Silicon Graphics, and personal computers. Although replaced by a single-purpose subroutine herein, the general-purpose subroutines were most helpful in a first version of this program. The suggestions and patience of Donald R. Wheeler, Pierre Steinmann, Douglas Jayne, and Frank S. Honecy of NASA Lewis while the computers were tied up are appreciated. The author would like to thank VG Scientific for providing extensive information about their data file header format.

Lewis Research Center
National Aeronautics and Space Administration
Cleveland, Ohio, July 13, 1989

APPENDIX—DESCRIPTION OF DATA FILE HEADER

In the following description of the data file header, data values are presented in hexadecimal (base 16 numbers) form. All hexadecimal numbers are underlined for clarity. Each hexadecimal digit represents four bits, requiring then only a two-digit hexadecimal number for each byte. The following is a comparison of equal values in hexadecimal (base 16), decimal (base 10), octal (base 8), and binary (base 2) number representations:

Hexadecimal:	1	2	3	4	5	6	7	8
Decimal:	1	2	3	4	5	6	7	8
Octal:	1	2	3	4	5	6	7	10
Binary:	0001	0010	0011	0100	0101	0110	0111	1000

Hexadecimal:	9	A	B	C	D	E	F	10
Decimal:	9	10	11	12	13	14	15	16
Octal:	11	12	13	14	15	16	17	20
Binary:	1001	1010	1011	1100	1101	1110	1111	10000

Sequentially, from the beginning of the data file:

Block 1

Bytes	Description
1-2	FF FF; two-byte integer, value always equals −1, Check_Word1
3-4	FF FF; two-byte integer, value always equals −1, Check_Word2
5-6	01 00; two-byte integer, with value of 1; version number, This_Version
7-8	03 00; two-byte integer, with value of 3; number of blocks in this file header, File_Header_Size
9-20	00 00; six two-byte integers, each with value of 0, Spare_Array
21-22	01 00; two-byte integer, with value of 1; number of spectral regions up to the maximum (32), always equals 1 as written by this program, No_of_Regions
23-24	two-byte integer number giving the number of 512-byte data blocks in the first region (128 data values per block), Region_Size
25-26	01 00; two-byte integer, value always equal to 1 in this program; number of blocks in region descriptor, Region_Header_Size
27-28	01 00; two-byte integer, value always equal to 1 in this program, Region_Header_Id
29-34	repeat of bytes 23−28 but for second data region, all values equal to 0 for one-region file created by this program
35-214	repeat of bytes 23−28 but for data regions 3 through 32; all values equal to 0 for the one-region file created by this program
215-216	01 00; two-byte integer, value always equal to 1 in this program; number of depths at which spectra exist, No_of_Levels
217-218	01 00; two-byte integer, value always equal to 1 in this program, No_of_Stages

Additional byte values as follows:

219-220	31 00
221-236	02 00 01 00 26 20 20 20 AE 9C CA 7F 00 00 2E 87; first element of Index_Entry_Array
237-252	03 00 01 00 27 20 20 20 AE 9C CA 7F 00 00 2E 87; second element of Index_Entry_Array
253-254	04 00; beginning of third element in array
255-256	two-byte integer giving the total number of 512-byte data blocks in the entire file, including all regions, same as bytes 23 and 24 above for the one-region files written by this program
257-268	21 20 01 00 01 80 01 00 00 00 01 84
269-284	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 53
285-476	repeat of bytes 269 to 284 twelve more times
477-512	FF FF; repeated values of −1

Block 2

Bytes	Description
1-2	05 00; two-byte integer, value always equals 5, identifies this as the Region Name Information block
3-17	up to 15-byte-long region name for first region using standard ASCII characters, unused bytes equal to the ASCII "space" character, hexadecimal 20
18-482	20; region names for regions 2 through 32, unused in this program, each 15 bytes in length, each byte equal to the 'space' character

Additional byte values as follows:

483-496	01 00 56 A2 40 82 50 53 20 20 20 20 20 20
497-512	20 20 20 20 20 20 01 00 21 20 20 20 AE 9C CA 7F

Block 3

Bytes	Description
1–2	02 00; two-byte integer, value always equals 2, identifies an Information block
3–4	two-byte integer, day of the month data taken
5–6	two-byte integer, day of the week data taken
7–8	two-byte integer, month data taken
9–10	two-byte integer, year data taken
11–12	two-byte integer, hour of the day data taken, military format from 0 to 23 hours
13–14	two-byte integer, minutes data taken
15–16	two-byte integer, seconds data taken
17–56	first 40-byte comment line, standard ASCII characters, unused bytes equal to the "space" character, hexadecimal 20
57–96	20; 40 ASCII "space" characters
97–136	second 40-byte comment line, standard ASCII characters, unused bytes equal to the "space" character, hexadecimal 20
137–176	20; 40 ASCII "space" characters
177–216	third 40-byte comment line, standard ASCII characters, unused bytes equal to the 'space' character, hexadecimal 20
217–482	20; ASCII "space" characters

Additional byte values as follows:

483–496	01 00 56 A2 40 82 50 53 20 20 20 20 20 20
497–512	20 20 20 20 20 20 01 00 21 20 20 20 AE 9C CA 7F

Block 4

Bytes	Description
1–2	01 00; two-byte integer, value always equals 1, identifies block as Data Header, occurs at beginning of each data region in multiregion data file, occurs only once in data files written by this program
3–4	00 00; two-byte integer, value always equals 0 here, identifies data as a spectrum
5–6	two-byte integer, gives the number of data points in the data set, Number_of_Channels
7–10	four-byte REAL*4 number, starting electron energy of spectrum, X_Start
11–14	four-byte REAL*4 number, ending electron energy of spectrum, X_End
15–18	four-byte REAL*4 number, minimum data value in any channel of the spectrum, Y_Minimum
19–22	four-byte REAL*4 number, maximum data value in any channel of the spectrum, Y_Maximum
23–26	four-byte REAL*4 number, energy increment between data points, X_Step
27	01; one-byte integer, value always equals 1 in this program, indicates that the X-axis values are "binding energy", X_Axis_Units
28	05; one-byte integer, value always equals 5 in this program, indicates that the Y-axis values (the data) are "Counts"

Note that values for the X- and Y-axis units are

00	Kinetic_eV (kinetic energy in electron volts)
01	Binding_eV (binding energy in electron volts)
02	AMU (atomic mass units)
03	Seconds
04	Degrees
05	Counts
06	Count_eV_per_seconds
07	CPS (counts per second)

29–30	01 00; two-byte integer, value always equals 1 in this program, No_of_Corresponding_Vars
31–34	four-byte REAL*4 number, value equals 0 here, Sensitivity_factor
35–38	four-byte REAL*4 number, value equals 0 here, Start_Profile_Range
39–42	four-byte REAL*4 number, value equals 0 here, End_Profile_Range
43–64	00 00; zero values
65–68	four-byte REAL*4 number, gives the dwell time per channel during each scan, in milliseconds, Dwell_Time
69–70	00 00; two-byte integer, value always equals 0 in this program, Signal_to_Noise
71–72	two-byte integer, gives the number of scans summed to produce the data set, No_of_Scans
73	00; one-byte integer, value always equals 0 in this program, indicates that the number of counts per channel were summed from scan to scan, Point_Repeat

Note that values for the mode of channel repetition are

00	Integrated
01	Averaged
02	Position_Sensitive

74	00; one-byte integer, value always equals 0 in this program, indicates that the data are XPS data, Type_of_Technique Note that values for the technique are 00 XPS 01 Auger 02 UPS 03 LEELS 04 ISS 05 SIMS
75	00; one-byte integer, value always equals 0 in this program, indicates that the analyzer mode is constant analyzer energy (CAE), Analyzer_Mode Note that values for the analyzer mode are 00 CAE (constant analyzer energy) 01 CRR (constant retarding ratio)
76	00; zero-value byte
77-80	four-byte REAL*4 number, analyzer energy in CAE mode, in electron volts, Analyzer_Value
81-84	four-byte REAL*4 number, work function of the analyzer, in electron volts from the vacuum level, Analyzer_Work_Function
85	one-byte integer, value equals either 0 or 1 in this program, indicates the excitation source, Source_Type Note that values for the type of source are 00 Al 01 Mg 02 Ag 03 Au 04 Zr 05 Unknown_Source 06 Helium_1 07 Helium_2 08 Electrons
86	00; zero-value byte
87-90	00; four-byte REAL*4 number, value always equals 0 in this program, Source_Strength
91-94	four-byte REAL*4 number, energy of the excitation source in electron volts, Source_Energy
95	00; one-byte integer, value always equals 0 in this program, indicates the type of signal collected, Signal_Mode_Type Note that values for the types of signal are 00 Pulse_Counting 01 Differential 02 Analogue
96	00; zero-value byte
97-100	00; four-byte REAL*4 number, value always equals 0 in this program, Modulation
101-104	00; four-byte REAL*4 number, value always equals 0 in this program, Analyzer_Polar_Angle
105-108	00; four-byte REAL*4 number, value always equals 0 in this program, Analyzer_Azimuth_Angle
109-112	00; four-byte REAL*4 number, value always equals 0 in this program, Sample_Polar_Angle
113-116	00; four-byte REAL*4 number, value always equals 0 in this program, Sample_Azimuth_Angle
117-120	00; four-byte REAL*4 number, value always equals 0 in this program, Sample_Rotation_Angle
121-124	00; four-byte REAL*4 number, value always equals 0 in this program, Sample_X
125-128	00; four-byte REAL*4 number, value always equals 0 in this program, Sample_Y
129-132	00; four-byte REAL*4 number, value always equals 0 in this program, Sample_Z
133-136	00; four-byte REAL*4 number, value always equals 0 in this program, Target_Bias
137-140	00; four-byte REAL*4 number, value always equals 0 in this program, Sample_Charging
141-155	up to 15-byte-long region name using standard ASCII characters, unused bytes equal to the ASCII "space" character, hexadecimal 20
156-195	20; 40-byte-long ASCII character label, all bytes equal to the "space" character in this program, hexadecimal 20, Label1
196-235	20; 40-byte-long ASCII character label, all bytes equal to the "space" character in this program, hexadecimal 20, Label2
236	00; zero-value byte
237-244	00; two four-byte REAL*4 numbers, values always equal 0 in this program, Label1_Position
245-252	00; two four-byte REAL*4 numbers, values always equal 0 in this program, Label2_Position
253	00; one-byte integer, value always equals 0 in this program, indicates the type of profile taken, Profile_Type Note that values for the types of profile are 00 Non_Profile 01 Cyclic_Etch_Time 02 Continuous_Etch_Time 03 Cyclic_Time

NASA
National Aeronautics and
Space Administration

Report Documentation Page

1. Report No. NASA TP-2957	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Fortran Program for X-Ray Photoelectron Spectroscopy Data Reformatting		5. Report Date November 1989
		6. Performing Organization Code
7. Author(s) Phillip B. Abel		8. Performing Organization Report No. E-4867
		10. Work Unit No. 506-43-11
9. Performing Organization Name and Address National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Paper
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546-0001		14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

A Fortran program has been written for use on an IBM PC/XT or AT or compatible microcomputer (personal computer, PC) that converts a column of ASCII-format numbers into a binary-format file suitable for interactive analysis on a Digital Equipment Corporation (DEC) computer running the VGS-5000 Enhanced Data Processing (EDP) software package. The incompatible floating-point number representations of the two computers were compared, and a subroutine was created to correctly store floating-point numbers on the IBM PC, which can be directly read by the DEC computer. Any file transfer protocol having provision for binary data can be used to transmit the resulting file from the PC to the DEC machine. The data file header required by the EDP programs for an x-ray photoelectron spectrum is also written to the file. The user is prompted for the relevant experimental parameters, which are then properly coded into the format used internally by all of the VGS-5000 series EDP packages.

17. Key Words (Suggested by Author(s)) X-ray; Photoelectron spectroscopy; VG Escalab; Surface properties; VAMAS		18. Distribution Statement Unclassified – Unlimited Subject Category 76	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No of pages 12	22. Price* A03

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA-Langley, 1989

National Aeronautics and
Space Administration
Code NTT-4

Washington, D.C.
20546-0001

Official Business
Penalty for Private Use, $300

BULK RATE
POSTAGE & FEES PAID
NASA
Permit No. G-27

NASA

POSTMASTER: If Undeliverable (Section 158 Postal Manual) Do Not Return

JUL 05 1989

E-4867 RM

NASA SCIENTIFIC AND TECHNICAL DOCUMENT AVAILABILITY AUTHORIZATION (DAA)

To be initiated by the responsible NASA Project Officer, Technical Monitor, or other appropriate NASA official for all presentations, reports, papers, and proceedings that contain scientific and technical information. Explanations are on the back of this form and are presented in greater detail in NHB 2200.2, "NASA Scientific and Technical Information Handbook."	[✓ Original] ☐ Modified	(Facility Use Only) Control No. ____ Date ____

I. DOCUMENT/PROJECT IDENTIFICATION (Information contained on report documentation page should not be repeated except title, date and contract number)

Title: FORTRAN Program for X-Ray Photoelectron Spectroscopy Data Reformatting

Author(s): Phillip Abel

Originating NASA Organization: Lewis Research Center

Performing Organization (if different): ____

Contract/Grant/Interagency/Project Number(s): 506-43-11

Document Number(s): [illegible]57 Document Date: ____

(For presentations or externally published documents, enter appropriate information on the intended publication such as name, place, and date of conference, periodical or journal title, or book title and publisher: NASA TP and Surface & Interface Analysis Journal

These documents must be routed to NASA Headquarters, International Affairs Division for approval. (See Section VII))

II. AVAILABILITY CATEGORY

IN-46
238727
10P

Check the appropriate category(ies):

Security Classification: ☐ Secret ☐ Secret RD ☐ Confidential ☐ Confidential RD [✓ Unclassified]

Export Controlled Document - Documents marked in this block must be routed to NASA Headquarters International Affairs Division for approval.

☐ ITAR ☐ EAR

NASA Restricted Distribution Document

☐ FEDD ☐ Limited Distribution ☐ Special Conditions-See Section III

Document disclosing an invention

☐ Documents marked in this block must be withheld from release until six months have elapsed after submission of this form, unless a different release date is established by the appropriate counsel. (See Section IX).

Publicly Available Document

[✓ Publicly available documents must be unclassified and may not be export-controlled or restricted distribution documents.]

☐ Copyrighted ☐ Not copyrighted

III. SPECIAL CONDITIONS

Check one or more of the applicable boxes in each of (a) and (b) as the basis for special restricted distribution if the "Special Conditions" box under NASA Restricted Distribution Document in Section II is checked. Guidelines are provided on reverse side of form.

a. This document contains:

☐ Foreign government information ☐ Commercial product test or evaluation results ☐ Preliminary information ☐ Information subject to special contract provision

☐ Other—Specify ____

b. Check one of the following limitations as appropriate:

☐ U.S. Government agencies and U.S. Government agency contractors only ☐ NASA contractors and U.S. Government agencies only ☐ U.S. Government agencies only

☐ NASA personnel and NASA contractors only ☐ NASA personnel only ☐ Available only with approval of issuing office; ____

IV. BLANKET RELEASE (OPTIONAL)

All documents issued under the following contract/grant/project number ____ may be processed as checked in Sections II and III.

The blanket release authorization granted ____ (Date) is:

☐ Rescinded - Future documents must have indivdual availability authorizations. ☐ Modified - Limitations for all documents processed in the STI system under the blanket release should be changed to conform to blocks as checked in Section II.

V. PROJECT OFFICER/TECHNICAL MONITOR

Stephen V. Pepper	5100	Stephen V. Pepper/sel	5/25/89
Typed Name of Project Officer/Technical Monitor	Office Code	Signature	Date

VI. PROGRAM OFFICE REVIEW [✓ Approved] ☐ Not Approved

M. S. Hirschbein	RM	[illegible]	6/24/89
Typed Name of Program Office Representative	Program Office and Code	Signature	Date

VII. INTERNATIONAL AFFAIRS DIVISION REVIEW

☐ Open, domestic conference presentation approved. ☐ Export controlled limitation is not applicable.

☐ Foreign publication/presentation approved. ☐ The following Export controlled limitation (ITAR/EAR) is assigned to this document: ____

☐ Export controlled limitation is approved.

International Affairs Div. Representative ____ Title ____ Date ____

VIII. EXPIRATION OF REVIEW TIME

The document is being released in accordance with the availability category and limitation checked in Section II since no objection was received from the Program Office within 20 days of submission, as specified by NHB 2200.2, and approval by the International Affairs Division is not required.

Name & Title ____ Office Code ____ Date ____

Note: This release procedure cannot be used with documents designated as Export Controlled Documents, conference presentations or foreign publications.

IX. DOCUMENTS DISCLOSING AN INVENTION

a. This document may be released on ____ (Date) Installation Patent or Intellectual Property Counsel ____ Date ____

b. The document was processed on ____ (Date) in accordance with Sections II and III as applicable. NASA STI Facility ____ Date ____

X. DISPOSITION

Completed forms should be forwarded to the **NASA Scientific and Technical Information Facility, P.O. Box 8757, B.W.I. Airport, Maryland 21240**, with either (check box)

☐ Printed or reproducible copy of document enclosed

☐ Abstract or Report Documentation Page enclosed. The issuing or sponsoring NASA installation should provide a copy of the document, when complete, to the NASA Scientific and Technical Information Facility at the above listed address.